What is claimed is:

1. A method for network packet processing comprises:

receiving network packets; and

operating on the network packets with a plurality of program threads to affect processing of the packets.

2. The method of claim 1 wherein operating comprises:

using at least one program thread to inspect a header portion of the packet.

3. The method of claim 1 wherein operating further comprises:

signaling by the at least one program thread that a packet header has been processed.

4. The method of claim 1 wherein the plurality of program threads are scheduler program threads to schedule task orders for processing and processing program threads that process packets in accordance with task assignments assigned by the scheduler program threads.

5. The method of claim 1 wherein each program thread writes a message to a register that indicates its current status.

6. The method of claim 5 wherein interpretation of the message is fixed by a software convention determined between a scheduler program thread and processing program threads called by the scheduler program thread.

7. The method of claim 5 wherein status messages include busy, not busy, not busy but waiting.

8. The method of claim 5 wherein a status message includes not busy, but waiting and wherein the status of not busy, but waiting signals that the current program thread has completed processing of a portion of a packet and is expected to be assigned to perform a subsequent task on the packet when data is made available to continue processing of the program thread.

9. The method of claim 5 wherein the register is a globally accessible register that can be read from or written to by all current program threads.

10. The method of claim 4 wherein scheduler program threads can schedule any one of a plurality of processing program threads to handle processing of a task.

11. The method of claim 10 wherein the scheduler program thread writes a register with an address corresponding to a location of data for the plurality of processing program threads.

12. The method of claim 11 wherein a selected one of the plurality of processing program threads that can handle the task reads the register to obtain the location of the data.

13. The method of claim 12 wherein the selected one of the plurality of processing program threads reads the register to obtain the location of the data and to assign itself to processing the task requested by the scheduler program thread.

14. The method of claim 12 wherein the selected one of the plurality of processing tasks reads the register to obtain the location of the data, while the register is cleared by reading the register by the program thread to assign itself to process the task.

15. The method of claim 13 wherein when another one of the plurality of processing program threads assignable to the task attempts to read the register after it has been cleared, it is

provided with a null value that indicates that there is no task currently assignable to the processing program thread.

16. A parallel hardware-based multithreaded processor for receiving network packets comprises:

a general purpose processor that coordinates system functions; and

a plurality of microengines that support multiple program threads, and operate on the network packets with a plurality of program threads to affect processing of the packets.

17. The processor of claim 16 wherein one of the plurality of microengines executes scheduler program threads and remaining ones of the microengines execute processing program threads.

18. The processor of claim 16 further comprising a global thread status register wherein each program thread writes a message to the global status register that indicates its current status.

19. The processor of claim 18 wherein interpretation of the message is fixed by a software convention determined between a

scheduler program thread and processing program threads called by the scheduler program thread.

20. The processor of claim 16 further comprising:

a read once register, wherein the scheduler program thread writes the read once register with an address corresponding to a location of data for the plurality of processing program threads and when a selected one of the plurality of processing program threads reads the register to obtain the location of the data, assigns itself to processing the task requested by the scheduler program thread, while the register is cleared by reading the register by the program thread.

21. The processor of claim 20 wherein when another one of the plurality of processing program threads assignable to the task attempts to read the read once register after it has been cleared, it is provided with a null value that indicates that there is no task currently assignable to the processing program thread.

22. An apparatus comprising a machine-readable storage medium having executable instructions for network processing, the instructions enabling the apparatus to:

receive network packets; and

operate on the network packets with a plurality of program threads to affect processing of the packets.

23. The apparatus of claim 22 wherein instructions to operate further comprise instructions to:

use at least one program thread to inspect a header portion of the packet.

24. The apparatus of claim 22 further comprising instructions to provide scheduler program threads to schedule task orders for processing and processing program threads to process packets in accordance with task assignments assigned by the scheduler program threads.

25. The apparatus of claim 22 wherein each program thread writes a message to a register that indicates its current status.

26. The apparatus of claim 25 wherein the register is a globally accessible register that can be read from or written to by all current program threads.

27. The apparatus of claim 22 wherein the scheduler program

thread writes a register with an address corresponding to a location of data for the plurality of processing program threads and a selected one of the plurality of processing program threads that can handle the task reads the register to obtain the location of the data, and clears the register after reading by the program thread.

28. The apparatus of claim 27 wherein when another one of the plurality of processing program threads assignable to the task attempts to read the register after it has been cleared, it is provided with a null value that indicates that there is no task currently assignable to the processing program thread.